Coursework 1

A Simple Perceptron Model For Points Classification

Kaiyuan Zhang[a,1]

[a]*SWJTU-LEEDS JOINT SCHOOL*

Abstract—To classify three distinct points with different labels, a simple perceptron model is implemented using Python. Using the number of misclassifications as the loss function, two different strategies are explored for updating the model weights. By visualizing the resulting decision regions, the effectiveness of each strategies is evaluated. The conclusion includes an analysis of which strategy performs best based on the observed results.

Keywords—*Classification, Perceptron*

1. Introduction

During the class, various methods for designing the loss function in a perceptron model were introduced. In this coursework, we chose the simplest approach—using the number of misclassified points as the loss function.

The main objective of this coursework is to observe how the decision region changes as the model weights are updated using different customized strategies. A total of two strategies were designed, including:

- Sequentially update weights for every misclassified sample in each epoch.
- Interactively let the user choose which misclassified sample to update in each epoch, and visualize the decision boundary after each update.

Finally, we compared the performance of the two strategies using the margin, defined as the minimum vertical distance from any data point to the decision region.

2. Methods

2.1. Building Dataset

To enable the model to read the data, a csv file named **test.data** was created.

There are three columns in the dataset. The first column is the x coordinate of the point, the second column is the y coordinate of the point, and the third column is the label of the point(1 or -1).

2.2. Model fitting

The first strategy was designed for more general situation, each epoch will traverse all samples and update the model weight for each sample, which means that there are multiple updates in one epoch.

The second strategy was designed for educational purpose. For each epoch, a text like this is provided:

```
Epoch 1: 3 misclassified points.
Misclassified points (index, x2, x1, label):
0: idx=0, x2=3, x1=3, label=1
1: idx=1, x2=4, x1=3, label=1
2: idx=2, x2=1, x1=1, label=-1
```

Then choose one point you'd like to update the weight model, after that you will see the decision region after this epoch. The loop will be over until there is no misclassified point.

2.3. Training Model

After training, the hyperplane obtained by the first strategy is:

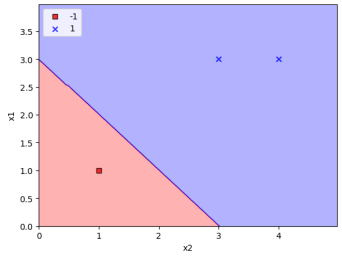

Figure 1. Decision region of first strategy.

The hyperplane obtained by the second interactive strategy turned out to have four main hyperplanes:

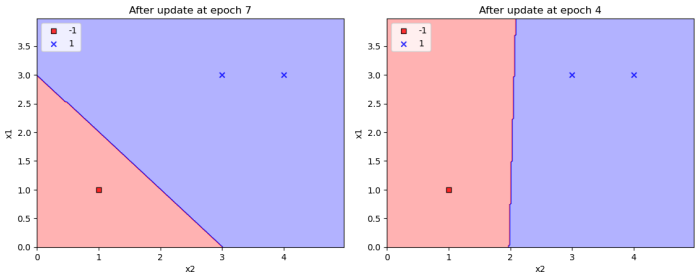

Figure 2

Figure 3

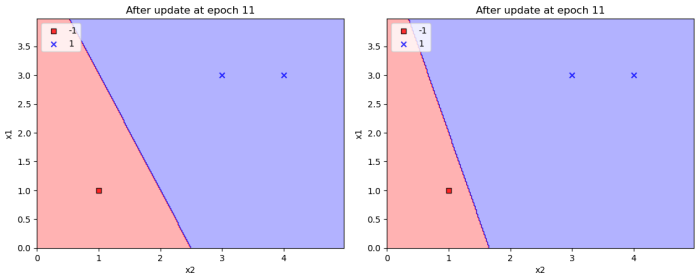

Figure 4

Figure 5

3. Results and Discussion

In order to evaluate which hyperplane is the best, two characteristics are selected to analyze the quality of the hyperplane:

- The number of epochs required to be able to classify the points
- The vertical distance from the nearest point to the decision region.

The optimal hyperplane determined according to these two criteria has the highest efficiency, the strongest generalization ability and the best noise resistance.

The results show that Figures 1 and 2 have margins of 0.7102183997942366, whereas Figure 3 achieves the higher margin (0.9386002176674437). The margins for Figure 4 and 5 are 0.8993398440374867 and 0.3122546977981159, respectively.

It turned out that Figure 3 has the best hyperplane, which only require 4 epoch to finish the classification and has the largest margin.

4. Appendix

The points selected for weight updates in each iteration, along with their sequencing, are detailed in the provided Jupyter notebook.